VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

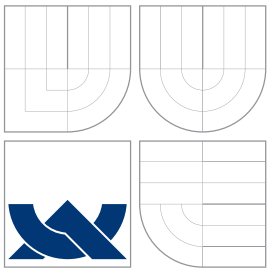
DATABÁZE XML PRO SPRÁVU SLOVNÍKOVÝCH DAT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

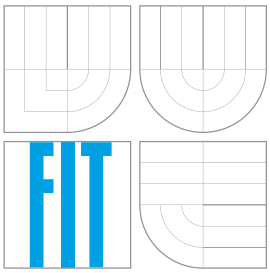
AUTOR PRÁCE
AUTHOR

MARTIN SKALICKÝ

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DATABÁZE XML PRO SPRÁVU SLOVNÍKOVÝCH DAT

XML DATABASES FOR DICTIONARY DATA MANAGEMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN SKALICKÝ

VEDOUCÍ PRÁCE
SUPERVISOR

doc. RNDr. PAVEL SMRŽ, Ph.D.

BRNO 2008

Abstrakt

Cílem praktické části této práce je převést nevalidní pseudoXML data do validního XML a dále pak provádět pokročilou validaci pomocí Schematronu. Teoretická část se týká popisu značkovacího jazyka XML. Dále se věnuje ukázkám a popisu dat a rozdílů standardů OLIF, ISLE/MILE a dalších. Část, kde je popsána implementace, vysvětluje problémy vzniklé při převodu do standartu a postup práce. V závěru práce je vyhodnocení statistik.

Klíčová slova

XML, slovníky, OLIF, převody, Schematron, ISLE/MILE, DTD., TBX, Saxon, statistiky slovníku, Python

Abstract

This Bachelor's thesis practical part's main goal is transformation of invalid pseudoXML data into valid XML and use Schematron for advance validation. Teoretical part is about XML markup language. Next part ilustrates and describes data storing in OLIF, ISLE/MILE etc. standards and differences between them. Part, where the thesis concentrates on implementation, describes problems and work progress. Last part is about statistic evaluation.

Keywords

XML, dictionary, OLIF, transformation, Schematron, ISLE/MILE, DTD., TBX, Saxon, dictionary statistics, Python

Citace

Martin Skalický: Databáze XML pro správu slovníkových dat, bakalářská práce, Brno, FIT VUT v Brně, 2008

Databáze XML pro správu slovníkových dat

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. RNDr. Pavla Smrže Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Skalický
13. května 2008

Poděkování

Zde bych chtěl poděkovat vedoucímu mé bakalářské práce doc. RNDr. Pavlu Smržovi Ph.D. za určování směru mé bakalářské práce a odbornou pomoc při její tvorbě.

Obsah

Kapitola 1

Úvod

Tato bakalářská práce se týká databází ve formátu XML pro správu slovníkových dat a je rozdělena na teoretickou a prakticou část.

Teoretická část práce se zaměřuje na slovníky, které jsou nejčastěji v knižní podobě a proto je potřeba řešit jejich uchovávání v univerzálním a dostupném formátu. Vhodné podmínky pro ukládání těchto informací poskytuje značkovací jazyk XML, který je blíže popsán v 2. kapitole práce. V tomto jazyce existují standardy pro ukládání slovníkových dat, jejich popisu a rozdílů mezi němi se věnuji v kapitole (3). Mezi takovéto standardy patří například OLIF (3.2. kapitola), TBX(3.1. kapitola) a další (3.3. kapitola).

Praktická část práce je implementace parseru v jazyce Python, jehož popis se nachází ve 4. kapitole. V této kapitole je vysvětlen postup pro převod nestandardního a nevalidního pseudoXML do standardu OLIF a problémy, které se při převodu vyskytly. Praktická část se zabývá také pokročilé validace dokumentu XML ve standardu OLIF, který jsem si vybral jako reprezentaci převedených dat, pomocí Schematronu (viz 4.3) a javovského programu saxon.

5. kapitola obsahuje popis získávání statistik za pomoci jazyku Python a parseru sax ze slovníku ve formátu OLIF a také některé získané statistiky z výsledného převedeného slovníku.

Kapitola 2

Značkovací jazyk XML

XML (eXtensible Markup Language, česky rozšiřitelný značkovací jazyk)[1] je značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C. I když se o XML mluví především v souvislosti s webovými aplikacemi (nástupce HTML)[9], je jeho rozšíření daleko větší. Díky tomu, že umožňuje popsat strukturu dokumentu z hlediska věcného obsahu i jednotlivých částí, používá se např. pro ukládání dat, výměnu informací mezi aplikacemi, tvorbě technické dokumentace atd.

2.1 Úvod do XML

Specifikace jazyku XML je *zdarma* dostupná každému, a to na stránkách konzorcia W3C.

V minulosti se používalo mnoho různých formátů pro uchovávání informací (např. DOC, XLS nebo PPT), které potřebovaly pro korektní zobrazení obsažených dat speciální software od různých distributorů. Dnes již tento způsob není vhodný, protože mnoho firem a uživatelů používá odlišné operační a informační systémy a není jisté, zda každý vlastní příslušné aplikace.

Vzniká tedy potřeba univerzálního, jednoduchého a volně přístupného formátu pro uchovávání dat. A přesně pro tento účel je vhodné XML.

2.1.1 Jazyková podpora

Výhodou XML je již od počátku podpora různých světových jazyků a znakových sad (implicitně je to ISO 10646). V jednom XML dokumentu můžeme mít texty v různých jazycích. Explicitně se může používat i jiné kódování např. utf8, windows-1250, iso-8859-2 atd. Informace o zvoleném kódování musí být v každém dokumentu přesně určena.

2.1.2 Obsah dokumentu XML

Význam jednotlivých částí dokumentu, napsaného pomocí XML, určují použité tagy (značky). Takto napsaný dokument obsahuje více informací než dokument, který nese

ještě navíc prvky vzhledu dokumentu, jako např. vzhled písma, odsazení, rozložení a podobně. XML nemá žádné prostředky pro určení vlastního vzhledu a o definici vzhledu se stará několik stylových jazyků. Tyto jazyky určují, jak se mají jednotlivé elementy zobrazit.

2.1.3 Transformace a zobrazení formátu XML

Jeden styl (viz kapitola 2.1.2) můžeme používat na dokumenty stejného typu, stejně tak je možnost pro jeden dokument určit více různých stylů. Existuje několik stylových jazyků. Díky využití ve webových aplikacích patří mezi nejznámější kaskádové styly (CSS). Ale lze je použít pouze pro jednoduché formátování dokumentu, které slouží k zobrazení dokumentu na obrazovce. Možností převodu formátu XML do jiného formátu je rodina jazyků XSL (eXtensible Stylesheet Language) (blíže v kapitole 2.5). Umožňuje dokument různě upravit a transformovat. Výsledkem může být HTML kód, PostScriptový soubor, zdrojový kód pro TEX a další.

2.2 Syntaxe XML dokumentu

2.2.1 XML značky

Samotné XML neobsahuje předdefinované značky (elementy, tagy), které tvoří většinu obsahu XML. Proto je třeba vytvořit vlastní elementy, jež budou popisovat a omezovat logické struktury dokumentu. Mohou obsahovat atributy, které dále popisují element (viz příklad 1).

Příklad 1 jednoduchý zápis elementů v XML 2.2.1

```
<Osoba>
   <Jmeno>Martin</Jmeno>
   <Prijmeni>Skalický</Prijmeni>
   <Narozen místo="Svitavy">15.04.1986</Narozen>
</Osoba>
```

Pro kontrolu správnosti takového XML dokumentu slouží parsery (kapitola 2.3) a validátory (kapitola 2.4).

2.3 Parsování XML

Když potřebujeme pracovat s XML dokumenty, tak není nutné psát vlastní analyzátor. Pro tento účel můžeme využít některé z již existujících parserů, např. SAX (kapitola 2.3.2) nebo DOM (kapitola 2.3.1). Samotný XML parser je program nebo programátorská knihovna, která se stará o načtení, nízkoúrovňovou syntaktickou analýzu XML dokumentu a jeho převod do infosetu. Infoset[20] popisuje informace,

jež lze získat (o uzlu, elementu, dokumentu, atributu a další).

Při parsování XML dokumentu se využívají dva hlavní přístupy. Prvním je událostmi řízené zpracování XML dokumentu. Tento způsob má dvě velké výhody:

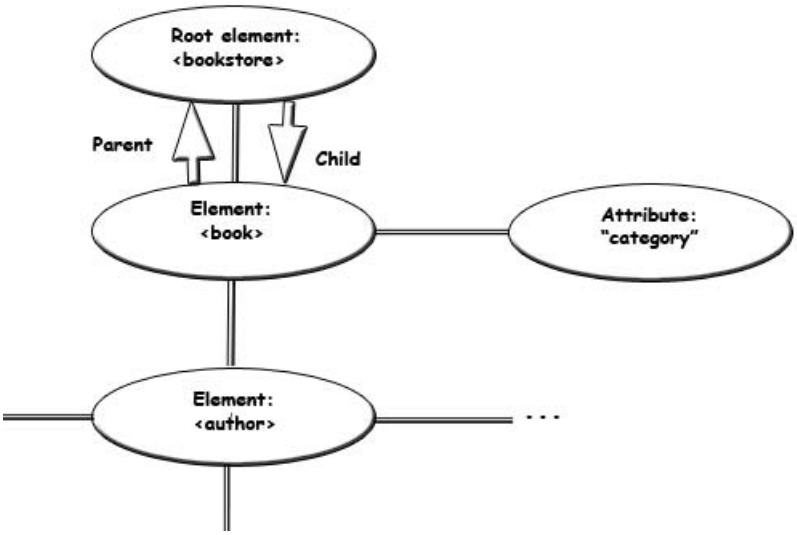
- je rychlý
- má malé paměťové nároky.

Naopak nevýhodou zůstává nutnost zpracovat XML dokument během jednoho sekvenčního průchodu.

Druhý přístup ke zpracování XML dokumentu představuje přístup založený na stromové reprezentaci dokumentu. Při tomto způsobu může - po načtení celého XML dokumentu (zaplatíme za to nižší rychlostí a velkou paměťovou náročností) - programátor kdykoliv přistupovat k libovolné části XML dokumentu. Každý objekt odpovídá jednomu uzlu ve stromu XML dokumentu a nabízí metody pro zjištění svého typu a hodnoty svých potomků a rodičů.

2.3.1 DOM

Nejznámějším rozhraním založeným na stromové reprezentaci XML dokumentu (viz kapitola 2.3.1) je DOM (Document Object Model)[19], který definuje standard pro přístup a práci s XML dokumenty. Tento standard vytvořilo konsorcium W3C a je zcela nezávislé na používaném programovacím jazyku. XML dokument je reprezentován objekty, jež zastupují jednotlivé prvky XML dokumentu (elementy, atributy, textový obsah atd.). Tyto objekty nabízí metody pro zjištění svých vlastností (typ, hodnota, rodiče, potomci a další).



Obrázek 2.1: Ukázka stromové struktury DOM

Strom dokumentu lze procházet libovolně a opakovaně, díky čemuž je zpracování XML dokumentu velmi jednoduché.

2.3.2 SAX 1.0

Asi nejznámější rozhraní používající událostmi řízený přístup je SAX[12] (Simple API for XML). SAX není standard W3C, ale vytvořila ho skupina lidí kolem konference XML-DEV. Původní návrh SAXu je pro programovací jazyk Java, ale jeho implementace se rozšířila pro mnoho dalších jazyků například Python. Oproti přístupu DOM je SAX vhodnější na velké dokumenty, protože je nemusí ukládat v paměti. Jeho pomocí se můžeme zaměřit pouze na část dokumentu, se kterou potřebujeme pracovat a vytvořit si ji jako strom. Nevýhody SAXu spočívají v tom, že si nemůžeme dokument upravit a nelze se vrátit na již zpracovaná data.

2.4 Struktura a validace dokumentu XML

Jazyk XML dovoluje vytváření vlastních značek (tagů), a proto také nastává potřeba popsat strukturu dokumentu XML. K tomuto účelu slouží schémata. V takovém schématu pak bude popsáno, jak bude vypadat struktura a omezení našeho nového značkovacího jazyku. Výhoda formalizované specifikace je v tom, že znemožňuje různé interpretace významu položek.

Schéma jednoznačně definuje, jak má XML dokument vypadat a proto ho lze využít pro validaci (ověření správnosti), což je hlavní použití schémat. *Validace* je proces, při kterém je ověřena správnost dodržení popisu schématu. Tato vlastnost se hodí, když si potřebujeme ověřit, jestli je dokument, který nám poslal kolega, v dohodnutém formátu. Validace ulehčuje práci i aplikacím, které nemusí provádět kontrolu požadovaného dokumentu. O tento úkol se stará právě validace.

Některé jazyky pro popis schématu dokumentu dokonce umožňují určit požadovaný typ dat jednotlivých elementů a atributů, jako řetězec, číslo, datum a další. Během validace se pak jednotlivým částem přiřazuje jejich datový typ, což je výhodné, protože aplikace pak nepracuje jenom s řetězci (jak je běžná praxe), ale s určenými datovými typy.

2.4.1 DTD

DTD (Document Type Definition)[18] je starší a podporováno množstvím aplikací. Pokud je specifikováno DTD, je možné automaticky kontrolovat, jestli je XML podle jeho předpisu. Tuto práci provádí parser (viz příklad 2.3). DTD není jediným definičním jazykem pro XML. Jeho nevýhodou je, že nepodporuje možnost kontroly typu dat (čísla, měnové údaje, datum a čas) a má prakticky nulovou podporu jmenných prostorů (umožňuje jednoznačnou indentifikaci elementů) viz příklad 2.

Příklad 2 velice jednoduché DTD 2.4.1

```
<!DOCTYPE Osoba [
   <!ELEMENT Osoba(Jmeno, Prijmeni, Narozen)>
   <!ELEMENT Jmeno(#PCDATA)>
   <!ELEMENT Prijmeni(#PCDATA)>
   <!ELEMENT Narozen(#PCDATA)>
   <!ATTLIST Narozen místo CDATA #REQUIRED>
]>
```

Pomocí DTD byla vytvořena různá schémata, která definují značky (názvy elementů) pro konkrétní typy dokumentů. Příkladem může být starší verze slovníku OLIF, který definuje struktury pro ukládání slovíkových dat. K některým schématům jsou dodávany i XSL soubory pro další zpracovávání dat. Další vlastností XML je, že v jednom dokumentu můžeme používat najednou nezávisle na sobě několik druhů značkovaní pomocí jmenných prostorů (namespaces). To umožňuje kombinovat v jednom dokumentů několik různých definic ve formě DTD nebo schémat bez konfliktů v pojmenování elementů.

2.4.2 Schéma XML

Kvalitnější možností popisu schéma XML dokumentu je XML schéma (XML Schema Definition (XSD)), které už umožňuje pro jednotlivé elementy definovat jejich datový typ a je celé zapsáno syntaxí XML. Složité XSD soubory není však příliš dobré tvořit jinak než za pomoci specializovaných editorů schémat. XML schéma definuje:

- Místa v dokumentu, na kterých se mohou vyskytovat různé elementy.
- Které elementy jsou potomky jiných elementů.
- Atributy, jejich datové typy a hodnoty.
- Pořadí, počty, datové typy a hodnoty elementů.
- Zda element může být prázdný, nebo zda musí obsahovat text.

2.4.3 Relax NG

Navrhnutý sdružením OASIS (The Organization for the Advancement of Structured Information Standards) je v dnešní době ISO standardem. Je postavený na základech jazyků RELAX a TREX.

- RELAX (Regular Language for XML, Description XML) - jednoduchý jazyk založený na matematické teorii alejových automatů aplikovanou XML stromy.
- TREX je jazyk pro validaci XML dokumentů.

Relax NG (REgular LAnguage for XML Next Generation) [15] je momentálně oblíbenější a kvalitnější [2] podporou pro psaní schémat XML dokumentu. Relax NG je založen na vzorech a ne na datových typech jako třeba XSD. To znamená, že celé schéma je vzorem dokumentu, který se skládá ze vzorů pro elemety, atributy a textové uzly. Vlastnosti vzorů:

- Mohou být dále kombinovány do uspořádaných i neuspořádaných skupin.
- Mohou být volitelné a může u nich být určen počet opakování.

Díky těmto vlastnostem a solidnímu matematickému základu můžeme snadno a přehledně popsat i složité struktury dokumentu. Další vlastností, jež dělá Relax NG ještě o něco lepší, je zápis schématu v textové syntaxi, která je úspornější než zápis založený na XML, který lze pro zápis také použít.

Relax NG samo o sobě nepodporuje datové typy a pro jejich doplnění je potřeba využít rozšíření.

2.4.4 Schematron

Výše zmíněná schémata zvládají validaci XML dokumentu, ale jejich hlavní činností je popis struktury dokumentu XML. Jazyk Schematron je založen na zcela odlišném principu. Jeho pomocí lze zjistit přítomnost nebo absenci určitých vzorů v dokumentu. K zápisu vzorů pro kontrolu dokumentu XML se využívá jazyku XPath (viz 2.6).

Díky spojení Schematronu a XPathu, kdy získáváme jeho silné vyjadřovací prostředky, nám pro validaci stačí XSLT procesor, protože schematronové schéma lze převést na XSLT transformaci.

Struktura schéma je velice jednoduchá. Elementy schematronu patří do jmenného prostoru http://purl.oclc.org/dsdl/schematron. Kořenový element *schema*, stejně jako ostatní elementy, obsahuje několik vzorů *pattern*.

Každý vzor je složen z jednoho nebo více pravidel *rule*, které mají v atributu *context* vzor zapsaný v jazyce XPath, který ze vstupního dokumentu vybere uzly, jež se chápou jako aktuální uzly pro vyhodnocení XPath výrazů uvnitř pravidla.

V pravidle se pak používají elementy *assert report*, které mají připojen atribut *test* obsahující XPath výraz.

- *Assert* - ověřuje nesplnění zadaných podmínek (např. neexistuje určitý záznam, takže vypíše hlášení).
- *Report* - ověřuje splnění zadané podmínky (např. elementy obsahují součet, vyšší než je únosné).

Uvnitř textu validačního hlášení můžeme používat další elementy.

2.5 XSL transformace 1.0

XSL je jazyk založený na XML, který slouží k převodu XML dokumentů na dokumenty jiného typu nebo stejného typu, když je potřeba odstranit nekompatibilitu

např. mezi verzemi návrhů XML dokumentů.

XSLT [14] nabízí tři odlišné modely programování: Model založený na vzorech, procedurální model a deklarativní model.

2.5.1 Model založený na vzorech

Tento model je nejjednodušší a umožňuje vzít šablonu XML dokumentu a naplnit ji XSLT výrazy, které dále dynamicky naplní příslušná místa dokumentu odpovídajícím obsahem. Aby byl tento model použitelný pro náš učel tak musí:

1. Být dobře strukturovaný XML dokument.
2. Být specifikováno číslo verze XSLT (např. `xslt:version='1.0'`).

Tato transformace odpovídá situaci, kdy máme jednu šablonu, která obsahuje celý vzorový dokument jako výsledný element viz příklad 3

Příklad 3 převod do HTML pomocí modelu založeného na vzorech 2.5.1

```
<html xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
   xsl:version='1.0'
   xmlns:v1='urn:zamestnanci:v1'>
   <body>
      <h1>
         <xsl:value-of select="concat(/v1:zam/krestniJmeno, ' ', /v1:zam/prijmeni)"/>
      </h1>
      <h2>
         <xsl:value-of select='/v1:zam/pozice'/>
      </h2>
   </body>
</html>
```

2.5.2 Procedurální model

XSLT také umožňuje oddělit a zobecnit transformační logiku od šablony. Šablony mohou být volány jako funkce stejně jako v procedurálních programovacích jazycích (viz příklad 4)

Příklad 4 zápis Procedurálního modelu 2.5.2

```
<xsl:transform
   xmlns:v1='urn:zamestnanec:v1'
   xmlns:v2='urn:zamestnanec:v2'
   xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
   version='1.0'>
   <xsl:template name="vytvorJmeno">
      <jmeno>
         <xsl:value-of select="concat(/v1:zam/krestni, ' ', /v1:zam/prijmeni)" />
      </jmeno>
   </xsl:template>
   <xsl:template match="/">
      <v2:zamestnanec>
         <xsl:call-template name="vytvorJmeno"/>
      </v2:zamestnanec>
   </xsl:template>
</xsl:transform>
```

2.5.3 Deklarativní model

XSLT nabízí silný a pružný model, který je podobný deklarativním jazykům jako Prolog, Lisp a Scheme. Tento model je založen na asociaci šablon se vzory či pravidly. Při provádění transformace procesor nejdříve vyhledá šablonu se vzorem odpovídajícím kořeni vstupního stromu. Uvnitř šablony si už můžeme samy určit uzly, kterými bude procesor procházet.

XSLT definuje několik vestavěných šablon, které jsou součástí každého programu (pokud nejsou explicitně překryty). Tyto šablony mají na programovací model hluboký účinek. Vestavěná šablona `apply-templates` říká, aby zpracování pokračovalo na všech dětských uzlech.

Deklarativní model umožňuje rozdělit transformační logiku do více modulů. Vývojář se dále nemusí zabývat tím, kdy a jak je šablona volána, stačí deklarovat, že daná šablona se volá pro konkrétní uzel. S tímto přístupem se dá snadno vytvořit program pro transformaci extrémně složitých dokumentů (příklad jednoduchého dokumentu viz 5).

Příklad 5 zápis Deklarativního modelu 2.5.3

```
<xsl:transform
   xmlns:v1='urn:zamestnanec:v1'
   xmlns:v2='urn:zamestnanec:v2'
   xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
   version='1.0'>
   <xsl:template match="text()|@*" />
   <xsl:template match="pozice">
      <titul><xsl:value-of select="." /></titul>
   </xsl:template>
   <xsl:template match="krestni">
      <jmeno>
         <xsl:value-of select="concat(., ' ', following-sibling::prijmeni)" />
      </jmeno>
   </xsl:template>
   <xsl:template match="v1:zam">
      <v2:zamestnanec>
         <xsl:apply-templates select="*"/>
      </v2:zamestnanec>
   </xsl:template>
</xsl:transform>
```

Přestože transformace založené na vzorech dovolují výstup pouze ve formátu XML (případně správně strukturovaném HTML), zbylé dva přístupy umožňují výstup v XML, HTML nebo prostém textu.

2.6 Adresace v XML

XML stejně jako HTML umožňuje vytváření odkazů v rámci jednoho dokumentu i mezi dokumenty, má však více možností. Je možné vytvářet i vícesměrné odkazy, které spojují více dokumentů dohromady. Tvorba odkazů je popsána ve třech standardech – XLink, XPointer a XPath.

- XPath (XML Path Language) je jazyk, který vyhodnotí výrazy podle stromu XML dokumentu a vrátí odpovídající uzly (elementy, atributy atd.).
- XPointer (XML Pointer Language), je rozšířením XPath. Není nutné části dokumentu, na které chceme odkazovat, explicitně označovat pomocí návěstí jako v HTML.
- XLink (XML Linking Language) je samotný jazyk pro tvorbu odkazů. Jednotlivé dokumenty se určují pomocí jejich URL adresy, za kterou lze uvést ještě XPointer pro přesnější určení části dokumentu.

Kapitola 3

Slovníkové databáze

Slovník[21] je nejvýznamnějším, většinou abecedně řazeným zdrojem informací o slovní zásobě jazyka. Slovníky vysvětlují slova z více hledisek. Lexikografie je lingvistická disciplína zabývající se sestavováním slovníků. Slovníky se vyskytují tradičně v knižní podobě, z čehož vyplývá, že data v nich uložená nejsou přímo určena pro počítačové aplikace. S vývojem informatiky se však objevují i digitální slovníky, dostupné na CD nebo na internetu. Podle typu dělíme slovníky na:

- Výkladové (jednojazyčné) - jsou napsány celé v jednom jazyce, u každého slova lze nalézt informace ve stejném jazyku.
- Současného jazyka (významové, pravopisné, frazeologické, slovníky synonym, slovníky cizích slov atd.).
- Jednotlivých historických období a slovníky etymologické.
- Popisující slovní zásobu profesních skupin (např. Filosofický slovník, Defektologický slovník, Biblický slovník apod.).
- Speciální (retrográdní, frekvenční, valenční atd.).
- Překladové (vícejazyčné).

Počet slov ve slovnících se pohybuje mezi 10 000 - 60 000 hesel. A každý ví jak zdlouhavé může být nalézt požadovanou informaci ve velkém knižním slovníku. Proto je výhodnější používat aplikace, které pracují se slovníkovými databázemi, vyhledávající automaticky požadované informace. Pro takovou databázi může být velice dobře použitelný právě dokument XML. Níže budou popsány některé osvědčené standardy pro uchovávání slovníkových dat.

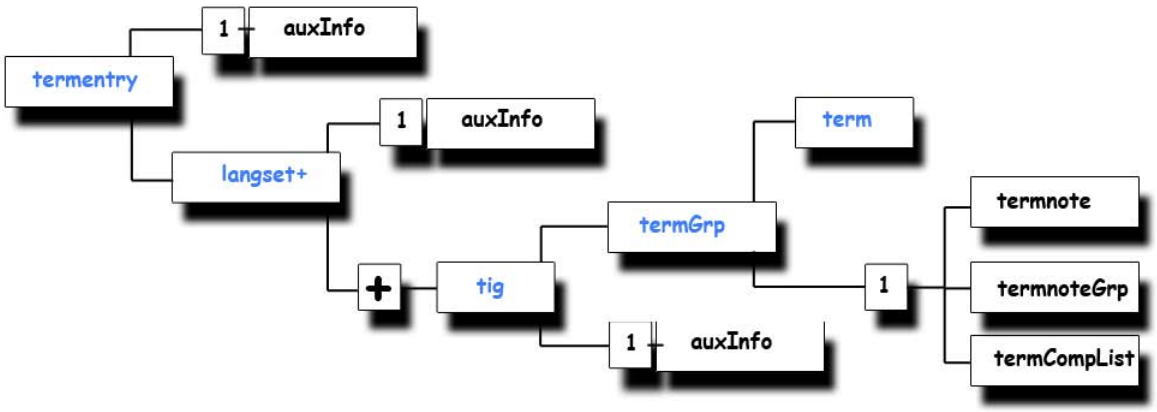
3.1 TBX

TBX(Term Base eXchange) [5] je jeden z prvních standardů pro ukládání slovníkových dat ve formátu XML a proto si zaslouží alespoň krátký popis. TBX popisuje terminologická slovníková hesla, která jsou založená na rozdílech mezi pojmy (významové

jednotky) a termíny (určují vstupy v různých jazycích) [16].

Rozdíl mezi pojmy a termíny je základním prvkem architektury TBX. Terminologická hesla jsou uspořádána v pojmy, které jsou základními významovými jednotkami a můžou obsahovat globální atributy jako oblast znalostí, příbuzné pojmy, definice, příklady, vzorové věty atd. Poté můžou být popsány jazykově příbuzným pojmenováním skupin informací o termínu ohraničujícím jednotlivé termíny.

Zápis výrazového pojmu je ukázán na obrázku 3.1.



Obrázek 3.1: Ukázka výrazového vstupu TBX

Výrazové pojmy tvoří jádro TBX dokumentu, který je zároveň XML dokumentem, obsahujícího následující složky:

- *Hlavičku* - popisuje dokument poskytnutím některých obecných a administrativních informací (obsah, status validace, kontakt, kódování, revize a další).
- *Tělo* - obsahuje sadu vstupů, jeden vstup na pojem z databáze. Tělo může mít úvodní a závěrečné elementy.

Popis elementů a jejich atributů v příkladu jednoduchého dokumentu TBX: 6. příklad:

- `<?xml ...`: Určuje, že se jedná o XML dokument verze 1.0
- `<! DOCTYPE martif ...`: Určuje, že XML dokument je validní podle specifikace TBX, která je popsána v TBXcoreStructV01.dtd (viz2.4.1), a podle XCS souboru v *encodingDesc* elementu.
- `<martif`: Jedná se o kořenový element. Jeho atributy určují:
  - type - o jaký se jedná dokument (v našem případě TBX)
  - xml:lang - jaký je výchozí jazyk dokumentu (v našem případě Angličtina podle ISO 639 zkratka 'en')

- `<martifHeader`: Obsahuje obecné informace o kolekci: Popis souboru (*fileDesc* element), který říká, že příklad je odvozený ze záznamu v Oraclu. TBX XCS viz výše.

- `<text> <body>`: Význam elementu *text*, do kterého náleží element *body*, je kompatibilita s Text Encoding Initiative guidelines [7]. Element *body* obsahuje kolekce pojmově orientovaných "Terminologických záznamů" `<termEntry>`.

- `<termEntry`: Každý *termEntry* element je jedna instance "Terminologického záznamu". Atribut *id* obsahující v našem případě 'eid-Oracle-67' se sestává z informace: *eid* `[indentifikátor záznamu]` + jméno databáze `[Oracle]` + sériové číslo záznamu (67)

- `<descrip type='subjectField'`: Element *descript* s atributem *type='subjectField'*, jehož obsah je schválený podle XCS souboru (viz výše), určuje oblast užívání záznamu.

- `<descrip type='definition'`: Tento element s atributem *type='definition'* obsahuje blížší popis pojmu.

- `<langSet xml:lang='en'>`: Určuje do jakého jazyku náleží následující terminologický záznam.

- `<tig><term>`: Obashuje termín ve výše učeném jazyce. Atribut *id* určuje: `[indentifikátor záznamu]` + jméno databáze `[Oracle]` + sériové číslo záznamu (67) + jazykový kód.

- `<termNote type='termType'`: Nese informaci o typu termínu. V našem případě plná forma.

- `</tig>`: Uzavření elementu *tig*.

- `</langSet>`: Uzavření anglické sekce *langSet*.

- `<langSet xml:lang='hu'>`: Tady začíná maďarská sekce překladu slova.

- `<tig>`: Obashuje maďarský termín pro výše uvedený anglický termín. Protože některé maďarské znaky nejsou podle standardu ISO 646, tak je záznam reprezentován několika Unikódovými hexa znaky. Správný maďarský výraz je pak: "Alfa simítási tényezõ".

- `</langSet>`: Konec maďarské jazykové sekce.

- `</termEntry>`: Konec výrazové sekce.

- `</body> </text>`: Ukončující tag pro terminologické záznamy, které v našem případě obsahovalo pouze jeden vstup.

- `</martif>`: Konec TBX dokumentu.

Příklad 6 zápis jednoduchého XML dokumentu podle standardu TBX 3.1

```
<?xml version='1.0'?>
<!DOCTYPE martif SYSTEM "TBXcoreStrucV01.dtd">
<martif type='TBX' xml:lang='en'>
   <martifHeader>
      <fileDesc>
         <sourceDesc>
            '<p>from an Oracle corporation termBase</p>'
         </sourceDesc>
      </fileDesc>
      <encodingDesc>
         <p type='DCSName'>TBXmasterXCSV01.XML</p>
      </encodingDesc>
   </martifHeader>
   <text> <body>
      <termEntry id='eid-Oracle-67'>
         <descrip type='subjectField'>manufacturing</descrip>
         <descrip type='definition'>A value between 0 and 1 used in ...</descrip>
         <langSet xml:lang='en'>
            <tig>
               <term id='tid-Oracle-67-en1'>alpha smoothing factor</term>
               <termNote type='termType'>fullForm</termNote>
            </tig>
         </langSet>
         <langSet xml:lang='hu'>
            <tig>
               <term id='tid-Oracle-67-hu1'>Alfa sim&#x00ED;t&#x00E1;si
 t&#x00E9;nyez&#x00F5; </term>
            </tig>
         </langSet>
      </termEntry>
   </body> </text>
</martif>
```

V době, kdy se začalo pracovat na výměnném formátu pro strojové překladové slovníky, bylo zřejmé, že TBX formát nebude schopen uspokojit požadavky pro tákové slovníky. Při výměně informací strojových překladů hledáme odpovědi na tři základní otázky:

1. Jaké záznamy vyměňujeme?
2. Jak může být popis jednotlivého záznamu vysvětlen?
3. Jak jsou mezi záznamy popsány vztahy?

Proč tedy není TBX vhodné?

- Není uspokojen požadavek na jazykovou výměnu, protože jsou vysvětleny jenom některé popisy slov jako slovní druh, rod, číslo. Ve standardu TBX také není žádná představa o skloňování, syntaktických typech, významových rysech atd., které jsou základní vlastností vyměny strojových překladů.

- Není jasné uspořádání jazykových popisů. Některé jsou spojeny s pojmovou úrovní (jako definice, příklady, vzahy) a další jsou součástí slovního druhu spojeného s termínovou úrovní. Z toho vyplývá nezbytná potřeba definování základních popisů (atributy a jejich platné hodnoty) pro lingvistické informace.

3.2 OLIF

3.2.1 Historie OLIFu

Následníkem standardu TBX se stal OLIF[3] (Open Lexicon Interchange Format), který odstraňuje nedostatky a doplňuje pravidla, se kterými se ve standardu TBX nepočítalo.

OLIF je otevřený slovníkový výměnný formát a nejdříve byl definován v EC projektu OTELO. Původně bylo zamýšleno zpřístupnit výměnu sad strojových překladových (dále jen SP) záznamů mezi SP automaty a uživateli. Jedním z důvodů bylo poskytnout výrazová data pro SP systémy jako *Logos* nebo *METAL*, což zahrnovalo samotný výměnný formát a převodníky z/do OLIFu poskytované SP automaty.

Pozdější verze výměnných formátů byly vyvinuty konzorciem OLIF za účasti hlavních SP poskytovatelů (Systran, Logos, SailLabs, linguatec), terminologických poskytovatelů a uživatelů (Trados, Microsoft, IBM, European Commission a další). Hlavní podnět byl(a stále je) veden od SAP(Systems - Applications - Products in data processing). Současné verze přidaly hlavičkovou strukturu, jako je tomu u TBX, a poskytují vícejazyčné ontologie (výslovný popis určité problematiky), lepší strukturování XML, několik nástrojů a podpůrných komponent [16].

3.2.2 Metamodel standardu OLIF

Základní rozhodovací architektura OLIFu je založená na pojmench (základní jednotka je sémantická entita). Pojmy v OLIFu, narozdíl od TBX konceptů, jsou definovány pro daný jazyk. Mezi pojmy jsou odkazy, z jednoho bodu určitého pojmu do druhého, tyto odkazy můžou být monolinguální (na základě slovníkových vztahů) nebo multilinguální (na základě překladů). Ve výsledku může být metamodel OLIFu chrakterizován:

- Jako založený na pojmech, které jsou monolinguální a mají jazykové vysvětlivky.

- Jako multilinguální (můžou zde být odkazy z pojmu na více cílových uzlů), ale přímý (odkazy mají zdroj a cíl a nejsou snadno zpětně převedeny).

3.2.3 Dokument XML ve standardu OLIF

Charakteristika klíčů (Key)

První věcí je charakteristika záznamů ve výměnném formátu. Vstupy ve formátu OLIF jsou chrakterizovány čtyřmi typy informací:

1. Kanonická forma (popis rozdílů mezi slovy např. Bank a Bank (anglicky banka nebo třeba břeh řeky)).
2. Jazyk (jazyk ve kterém je definice záznamů).
3. Výslovnost.
4. Sémantické popisy (poskytuje popis rozdílů v praktických situacích).

Ukázka a popis dokumentu XML podle OLIF

Kořenovým elementem u formátu OLIF je element *olif*. Stejně jako u standardu TBX je dokument OLIFu rozdělen na dvě sekce.

1. *Header* - hlavička obsahující obecné informace o slovníku.
2. *Body* - tělo dokumentu, které obsahuje slovníková data. Jaká data bsahuje se pokusím vysvětlit právě v této kapitole.

Tělo dokumentu čili tag *body* musí obsahovat 1-N elementů typu *entry*. Dále pak obsahuje:

- *Jednojazyčné informace (Monolingual Information)* - skupina jednojazyčných informací zahrnuje všechny datové kategorie, které jsou původně jednojazyčné:
  - *Kategorie klíčů* viz 3.2.3.
  - *Administrativní datové kategorie.*
  - *Morfologické datové kategorie* - jako struktura, skloňování, pád, číslo, osoba a čas.
  - *Sémantické datové kategorie* - definice, sémantika, typ a pohlaví.
  - *Obecné datové kategorie* -příklad, poznámka.
- *Odkazované informace (CrossReference informations)* - jsou skupiny informací o příbuzných záznamech ve stejném jazyce jako je překládané slovo a obsahují tyto kategorie:
  - *Kategorie klíčů* viz 3.2.3.
  - *Druh vztahu* - podle doporučení z EuroWorldNetu to můžou být asociativní výraz, hypernym (obecné slovo pro více specifických slov jako dopravní prostředek je obecné slovo pro automobil, autobus nebo třeba letadlo), meronym (označuje základní část nebo člena něčeho např. prst je meronym ruky, kolo je meronym automobilu).

- *Informace o překladu (Transfer informations)* - jsou skupiny informací o záznamech v jazyce, který je odlišný od vstupního jazyku. OLIF podporuje přímé vícejazyčné překlady (dvojjazyčné překlady jsou speciální případ). Obsahuje kategorie pro:
  - *Kategorie klíčů* viz 3.2.3.
  - *Typ rovnosti* (např. plná (full)).
  - *Omezení*, které určují podmínky, za kterých je překlad platný.
  - *Datové skupiny*, které vysvětlují strukturální změny pro daný překlad. (zhasnout `->` switch off).

Příklad záznamu jednoho slova ve formátu OLIF

Ukázka překladu jednoho slova za použití OLIFu v příkladu 7, kdy:

- element *mono* obsahuje informace o překládaném slově v elementech:
  - *canForm* obsahuje kanonický tvar překládaného slova
  - *language* jazyk, do kterého slovo náleží
  - *subjField* oblast výskytu slova
  - *pronunciation* výslovnost slova
- element *transfer* obsahuje informace o překladu slova v elementech:
  - *canForm* obsahuje kanonický tvar slova
  - *language* jazyk, do kterého slovo náleží
  - *subjField* oblast výskytu slova

Příklad 7 Překlad jednoho slova ve slovníku OLIF 3.2.3

```
<entry>
   <mono>
      <keyDC>
         <canForm>abbatial</canForm>
         <language>en</language>
         <subjField>general</subjField>
         <pronunciation>{beišl</pronunciation>
      </keyDC>
   </mono>
   <transfer>
      <keyDC>
         <canForm>opatský</canForm>
         <language>cs</language>
         <subjField>general</subjField>
      </keyDC>
   </transfer>
<transfer>
    <keyDC>
         <canForm>abatyšský</canForm>
         <language>cs</language>
         <subjField>general</subjField>
      </keyDC>
   </transfer>
</entry>
```

Příklad záznamu slova ve slovníku OLIF s odkazem na další slovo

V příkladu 8 bude ukázáno jak vypadá překlad slova s odkazem na další slovo ve stejném jazyce a s pomocnými informacemi. Vysvětlení elementů, které neobsahuje příklad 7:

- Element *generalDC* obsahuje informace o tom, jak slovo použít nebo poznámku překladatele.
- Element *crossRefer* nese informace o odkazu na varianty překládaného slova:
  - Element *crLinkType* určuje, o jaký typ odkazu jde. V našem případě headword (hlavní slovo kanonické formy).
- Element *monoMorph* - uvádíme informace o morfologii slova. V *morphStruct* je uvedena stavba slova, inflexní vzor, rod, číslo, vid a další.
  - Element *number* - informace o čísle.
- Element *crLinkType* - obsahuje roli, kterou slovo hraje v kontextu, např. *orthvariant* že jde o jinou variantu slova nebo *headword*, že jde o slovo, ze kterého vychází např. fráze.

Příklad 8 Překlad slova s odkazem na další slovo v OLIFu 3.2.3

```
<entry>
   <mono>
      <keyDC>
         <canForm>A</canForm>
         <language>en</language>
         <ptOfSpeech>noun</ptOfSpeech>
         <subjField>general</subjField>
         <pronunciation>ei</pronunciation>
      </keyDC>
      <monoDC>
         <monoMorph>
            <number>pl</number>
         </monoMorph>
      </monoDC>
      <generalDC>
         <usage>písmeno</usage>
      </generalDC>
   </mono>
   <crossRefer>
      <keyDC>
         <canForm>a</canForm>
         <language>en</language>
         <ptOfSpeech>noun</ptOfSpeech>
         <subjField>general</subjField>
      </keyDC>
      <crLinkType>orth-variant</crLinkType>
   </crossRefer>
   <transfer>
      <keyDC>
         <canForm>A</canForm>
         <language>cs</language>
         <ptOfSpeech>noun</ptOfSpeech>
         <subjField>general</subjField>
      </keyDC>
   </transfer>
</entry>
```

Příklad fráze ve slovníku OLIF

V posledním, (9). příkladu, který se týká slovníku OLIF uvedu, jak vypadá fráze a její překlad. Navíc oproti minulým příkladům se objevuje element *monoAdmin*, který obsahuje vlastníka slova, lokalitu používání slova a podobně. Například v elementu *entryFormation* uvedený typ záznamu (např. fráze).

Příklad 9 Fráze ve slovníku OLIF 3.2.3

```
<entry>
   <mono>
      <keyDC>
         <canForm>an A1 population</canForm>
         <language>en</language>
         <ptOfSpeech>other</ptOfSpeech>
         <subjField>general</subjField>
      </keyDC>
      <monoDC>
         <monoAdmin>
            <entryFormation>phr</entryFormation>
         </monoAdmin>
      </monoDC>
   </mono>
   <crossRefer>
      <keyDC>
         <canForm>A</canForm>
         <language>en</language>
         <ptOfSpeech>noun</ptOfSpeech>
         <subjField>general</subjField>
      </keyDC>
      <generalDC>
         <usage>hovor.</usage>
      </generalDC>
      <crLinkType>headword</crLinkType>
   </crossRefer>
   <transfer>
   <keyDC>
      <canForm>obyvatelstvo v dokonalém zdravotním stavu</canForm>
      <language>cs</language>
      <ptOfSpeech>other</ptOfSpeech>
      <subjField>general</subjField>
   </keyDC>
   </transfer>
</entry>
```

3.3 Další standardy

3.3.1 ISLE/MILE

ISLE (Internation Standards for Language Engineering)[22] je název projektu a celá sada organizovaných aktivit týkajících se oblasti terminologie lidského jazyka (HLT - Human Language Technology). ISLE spadá pod iniciativu EAGLES (Expert Advisory

Group for Language Engineering Standards), která vykazuje úspěšný vývoj a značné množství doporučení a standardů. ISLE je zaměřeno na 3 hlavní oblasti:

- mnohojazyčné slovníky
- přirozená interakce a multimodalita
- vyhodnocení HLT systémů

Standard MILE (Multilingual ISLE Lexical Entry)[6] je výsledkem výzkumu založeného právě na EAGLES/PAROLE. Představuje reprezentaci multilinguálních informací v konstrukci vrstvených lexikálních reprezentačních standardů. Morfologická skladba je definována podle projektu PAROLE, sémantika podle projektu SIMPLE a vícejazyčnost podle projektu ISLE.

Rozdíl mezi MILE a OLIF

Narozdíl od standardu OLIF, MILE pokrývá nejen informační položky, které jsou dostupné ve dnešních SP slovnících, ale hodlá představovat kompletní lexikální popis včetně sémantické reprezentace vícejazyčnosti. MILE není výměnný standard, ale je reprezentační standard a může být převeden do mnoha různých výměnných formátů (dokud budou mít potřebnou vyjadřovací sílu podporovat MILE). Záznamy v MILE můžou definovat pořadí popisovacích podmínek k vyjádření specifického omezení sady v překladovém kontextu bez vlivu na monolinguální vyjádření záznamu.

Příklad záznamu v MILE

Lexikální záznam v MILE je ideální strukturou pro převody RDF(Resource Definition Framework)/OWL(Ontology Web Language). To spočívá v hierarchii lexikálních objektů, které jsou postavené na vrstveném způsobu kombinováním základních datových kategorií pomocí jasně daných vztahů.

Uvnitř každé vrstvy slovníku MILE jsou dva typy objektů:

1. Lexikální třída MILE(LTM).
   - Hlavní stavební blok lexikálních záznamů.
   - Formalizuje základní lexikální pojmy podle projektu ISLE.
   - Lexikální model MILE(LMM) Definuje každou třídu specifikováním jejích atributů a vztahů mezi nimy.
   - Popisuje pojmy jako syntaktická vlastnost, synset atd.
2. Lexikální operace - speciální lexikální záznamy, které dovolují uživatelům popsat podmínky a vykonávat komplexní operace nad lexikálními vstupy.

Příklad plného záznamu ve slovníku MILE (10). Využívá vyčíslovací třídy v LDCR pro *SynFeatureName* a *SynFeatureValue*. V tomto případě LDCR pouze poskytuje uzavřený seznam možných hodnot, ze kterých musí být určená hodnota vybrána.

Příklad 10 Příklad plného záznamu ve slovníku MILE 3.3.1

```
<?xml version="1.0"?>
<!-- Sample ISLE lexical Entry for EAT (transitive), SynU only
      Abbreviated syntax version using no pre-defined objects
      2002/10/23 Author: Nancy Ide -->
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:mlc="http://www.cs.vassar.edu/ ide/rdf/isle-schema-v.6#"
      xmlns="http://www.cs.vassar.edu/ ide/rdf/isle-schema-v.6#">
<Entry rdf:ID="eat1">
   <!-- The SynU for eat1 -->
   <hasSynu rdf:parseType="Resource">
      <SynU rdf:ID="eat1-SynU">
      <example>John ate the cake</example>
      <hasSyntacticFrame>
         <SyntacticFrame rdf:ID="eat1SynFrame">
            <hasSelf>
               <Self rdf:ID="eat1Self">
                  <headedBy>
                     <Phrase rdf:ID="Vauxhave">
                        <hasSynFeature>
                           <SynFeature>
                              <hasSynFeatureName rdf:value="aux"/>
                              <hasSynFeatureValue rdf:value="have"/>
            </SynFeature></hasSynFeature></Phrase></headedBy></Self></hasSelf>
            <hasConstruction>
               <Construction rdf:ID="eat1Const">
                  <slot>
                     <SlotRealization rdf:ID="NPsubj">
                        <hasFunction rdf:value="Subj"/>
                           <filledBy rdf:value="NP"/>
                  </SlotRealization></slot>
                  <slot>
                     <SlotRealization rdf:ID="NPobj">
                     <hasFunction rdf:value="Obj"/>
                     <filledBy rdf:value="NP"/>
            </SlotRealization></slot></Construction></hasConstruction>
            <hasFrequency rdf:value="8788" mlc:corpus="PAROLE"/>
</SyntacticFrame></hasSyntacticFrame></SynU></hasSynu></Entry></rdf:RDF>
```

3.3.2 XLIFF

XLIFF [4] je dalším XML formátem pro slovníková data, který umožňuje zaměření se na překládaný text. Díky XLIFFu je lokalizační technika snažší: jak jednou převedete data ze svých zdrojů, tak už můžete jednoduše napsat nové nástroje pro práci s XLIFFem a nestarat se o původní formát. XLIFF také podporuje plný lokalizační proces, tím že poskytuje tagy a atributy pro shrnutí, status překladu jednotlivých řetězců atd.

Formát XLIFF se vyvíjel ve spolupráci více společností, ale nakonec se dostal pod křídla společnosti OASIS. XLIFF se zaměřuje na:

- Oddělení textu k lokalizaci od formátování.
- Zpřístupnění řady nástrojů k práci se zdrojovými řetězci a přidáváním dat o řetězci.
- Ukládání unformací, které jsou užitečné k podpoře lokalizačního procesu.

V nejjednodušší formě sestává XLIFF z jednoho nebo více souborových elementů. Každý z nich obsahuje sekce *header* (hlavička) a *body* (tělo). Header obsahuje stejně jako u předešlých formátů informace o projektu, autorovi atd. Body pak obsahuje elementy *trans-unit*, což jsou hlavní elementy dokumentu XLIFF, v nichž je uložen lokalizovatelný text a jeho překlady. Tyto elementy reprezentují segmenty (obvykle věty ze zdrojového souboru, které mohou být nezávisle přeloženy). Elementy *trans-unit* obsahují zdroj, cíl, alt-trans a další užitečné elementy.

Kapitola 4

Implementace

V této kapitole jsou popsány praktické části bakalářské práce. Vysvětlím, s jakými daty jsem pracoval a jak jsem je převáděl do slovníkového standardu OLIF. Následuje postup pokročilé validace dokumentu OLIF a na závěr shrnu s jakými problémy jsem se setkal.

4.1 Vstupní data

Vstupní data, se kterými jsem pracoval by se dala charakterizovat jako pseudo XML data. Jedná se o anglicko-český slovník pro písmena a-v rozdělený do 127 textových souborů, které jsem spojil do jednoho pro další jednodušší práci. Ukázka vzorku dat z původního textového souboru `a1_xml.txt` (viz 1).

```
<hwen1>abbreviation</hwen1>
<pronun1>[??bri:vi?eiš?n]</pronun1>
<sectn1>1</sectn1>
<hwecs1>z|krácení</hwecs1>
<sectn1>2</sectn1>
<hwecs1>zkratka @@@@@</hwecs1>
```

Ukázka 1: popis vstupních dat(4.1)

Jak můžeme vidět v ukázce, anglické slovo určené k překladu je logicky obsaženo v elementu **hewn1** (v našem případě se jedná o slovo *abbreviation*). Za tímto elementem následuje značka **pronun1** s výslovností, která není podle pravidel SAMPA [11], takže ji v programu převádím do tohoto formátu. Tag **sectn1** odděluje možnosti překladu anglického slova, v tomto případě se objevuje dvakrát, a to pro česká slova v elementu **hwecs1**.

Vstupní data obsahují mnohem více tagů, takže zde uvádím popis každého z nich:

- hwen1 - anglické slovo určené k překladu
- vwen1 - další tvar anglické slova

- pronun1 - výslovnost anglického slova
- hwecs1 - český překlad anglického slova
- phrwen1 - fráze v angličtině
- phrven1 - fráze v angličtině
- phrecs1 - česká fráze pro anglickou frázi uvedenou v *phrwen1* nebo *phrven1*
- examen1 - anglický příklad užití slova za pomoci fráze
- examcs1 - český překlad pro *examen1*
- compen1 - obsahuje usage pro anglické slovo
- undef1 - další role anglické fráze nebo slova
- hypodkaz - odkaz na slova stejného významu
- sectn1 - významové sekce pro anglická slova
- habbren1 - význam anglické zkratky
- undefi1 - nachází se před undef1 a říká, že undef1 je další tvar pro anglické slovo nebo frázi
- indcs1 - příklad užití pro české slova nebo oblasti užití slova
- regcs1 - regionální užití slova
- gram1 - slovní druh
- partp1 - odděluje příklady
- partc1 - také odděluje příklady
- hwsen - odděluje významy anglického slova

4.1.1 Statistiky ze vstupních dat

Statistiky ze vstupních dat jsem získával za pomoci jazyku Python. Tyto statistiky jsem využíval k určení hodnot, kterých nabývají elementy slovníku OLIF pro fixní hodnoty. Např. v elementu *gram1* je hodnota *adj:*, která říká, že se jedná o přídavné jméno. Také jsem potřeboval zjistit nejdelší vzdálenost významu slov pro funkci určování významu slov v parseru, kdy vím že význam slova je před slovem, takže hledám regulárním výrazem nejbližší význam před slovem a to od pozice: *vzdálenost = pozice slova - největší vzdálenost*. Jen pro zajímavost je v původních datech:

- Nejdelší záznam pro slovo *go* a má 102716 znaků (i se značkami).

- Je po rozdělení všech záznamů nesoucích informace o anglických slovech - 90177 různých slov.
- Nejčastějším slovem z těchto záznamů pak je slovo: ”the”, které je přítomno ve všech záznamech 37889 krát.
- Je po rozdělení všech záznamů nesoucích informace o českých slovech - 112355 různých slov.
- Nejčastějším slovem z těchto záznamů pak je slovo: ”se”, které je přítomno ve všech záznamech 21946 krát.

4.2 Parser pro převod dat do OLIFu

Pro převod dat z pseudo XML jsem využil již existujícího pythonovského parseru, který jsem upravil pro moje specifické vstupy. Dále jsem přidal funkce pro doplnění křížových odkazů, které tam chyběly, což jsou zkratky slov např. *AAA* znamená *Agricultural Adjustment Act*, funkci pro přidělení správné hodnoty usage a subjField k českým překladům slov a funkci pro vyhledání a určení další role fráze nebo slova uložené v tagu *undefi1*.

Také jsem program doplnil pro kontrolu existence anglických slov ve slovníku *ispell.words*. Prvním (nejjednodušším) způsobem bylo volání externích programů *cat a grep* pomocí pythonovského modulu *Popen*: `cat ispell.words| grep -E '^slovo'$`, což provede výpis všeho, co je v souboru *ispell.words* a následnou kontrolu existence slova pomocí regulárního výrazu. Toto řešení je velice neefektivní a pomalé, takže jsem se rozhodl tento soubor na začátku načíst do pole a vyhledávat v něm slovo pomocí funkce pythonovské struktury slovníku: `dict.has_key()`. Pokud slovo nebylo ve slovníku nalezeno tak ho zapíšu do souboru *notFoundEN.txt*.

Ostatní chyby a hlášení se zapisují do souboru *errors.txt*, můžou tam být například geografické užití slov bez ekvivalentu v poli geografických užití, nerozpoznaný slovní druh a další. Vzhledem k tomu, že v mých datech se nalézá více geografických užití než v původním skriptu, tak byla chybějící data doplněna. Jak již je napsáno výše, element `<indcs1>` obsahuje jak data pro *usage* tzn. užití slova, kde se může nacházet jakýkoliv řetězec, tak některé zkratky jako *práv.* což se dá použít v elementu `<subjField>`, který má omezený výběr hodnot např. pro práv. *law*. Problém je v tom, že tyto zkratky jsou zapsány v delším řetězci a odděleny `,;:`, takže pomocí regulárního výrazu tyto slova odděluji a vybírám z pole, kde jsem ručně zapsal možné zkratky pro zápis do `<subjField>`.

Další významný element, který nese informaci o slovním druhu, je `<ptOfSpeech>` a pole pro jeho hodnoty byla také doplněna. Pole s hodnotami pro `<geoUsage>` je také doplněná pro hodnoty z mých vstupních dat a to ve formátu mezinárodních poštovních zkratek [13]. Pokud je slovo užíváno ve více zemích tak přidávám mezi jednotlivá slova slovo *and* (např. *USA and AUS and NZL* což znamená, že je slovo typické pro Spojené státy americké, Austrálii a Nový Zéland.)

4.3 Schematron pro validaci výsledného XML dokumentu

Obecný popis nástroje schematronu je v kapitole 2.4.4. Já schematron používám na kontrolu hodnot v elementech OLIFu. A to konkrétně pouze v elementech, které obsahují fixní hodnoty [17]. V mém dokumentu OLIF to jsou:

- `<ptOfSpeech>` - obsahuje slovní druh (noun, verb, adj a další)
- `<subjField>` - v jaké oblasti se slovo využívá (economics, botany/zoology, law a další)
- `<number>` - číslo slova (sg(jednotné), pl, sgt, plt, du, invar a un)
- `<aspect>` - vid slova (simp, perf, imperf a další)
- `<crLink>` - vztah jednoho slova k druhému ve stejném jazyce (headword, orthvariant, abbreviation a další)

4.3.1 Programy pro spuštění schematronu

Pro zpracování schematronu se dají použít různé programy. Například používám program *XT* [10], což je program napsaný v jazyce Java, který umí XSL transformace. Je volně ke stažení. Pro správný chod stačí mít stažené nástroje pro Java Virtual Machine, nastavit si v souboru *build.bat* (pro Windows) nebo *build.sh* (pro Linux) cestu k adresáři, kde je umístěn Java Virtual Machine a pak už jenom spustit vybraný soubor, který vytvoří spustitelný program *XT* pro práci s předpisy ve schematronu. V další ukázce je příklad spuštění transformace pomocí programu XT, kdy rozdíl oproti příkladu saxonu (4.3.1) je ten, že používám styly pro transformaci výpisu chyb do souboru shcematron-errors.html, kde jsou odkazy na určité řádky, obsahující hlášku ze schématu, ve schematron-out.html.

Další zajímavou možností je wysiwyg editor *Oxygen XML*, který v sobě obsahuje i výše zmíněný saxon a další nástroje pro provádění XSL transformací. Můžou se v něm vytvářet dokumenty XML, XSL, SCM atd, obsahuje debuger a validátor dokumentu.

```
xt.exe schematron-predpis.xml schematron-report.xsl  xxx.xsl
xt.exe example.xml xxx.xsl schematron-errors.html
xt.exe example.xml verbid.xsl schematron-out.html
```

Ukázka 2: spuštění transformace pomocí XT

Já pro vyhodnocení předpisu schematronu používám saxon [8], což je XSLT procesor, který ze schematronového schématu vygeneruje další XSLT styl. Tímto stylem pak můžeme transformovat dokument, který chceme validovat. Výsledkem

transformace je pak seznam chyb. Ukázka spuštění programu pomocí saxonu (4.3.1), kdy přepínač *-o validuj.xsl* v prvním řádku znamená, že výstupní XSL soubor, který se použije pro transformaci na výpis chyb, se bude jmenovat validuj.xsl. Další parametry na prvním řádku jsou předpis pomocí schematronu a styl schematronu, který jenom vygeneruje výše zmíněný styl.
Výpis se provede do souboru `schematron_report.txt`.

```
java -jar ./Schematron/saxon/saxon9.jar -o ./Schematron/validuj.xsl \
          ./Schematron/schematron-predpis.sch \
          ./Schematron/iso_schematron_skeleton.xsl

java -Xms5m -Xmx2548m -jar ./Schematron/saxon/saxon9.jar
     -o schematron_report.txt out.xml ./Schematron/validuj.xsl
```

Ukázka 3: spuštění transformace pomocí saxonu

4.3.2 Předpis schematronu pro validaci

Každý dokument schematronu by měl začínat dekalrací XML:
`<?xml version="1.0" encoding="UTF-8"?>`
Dále by měl v kořenovém elementu obsahovat jmenný prostor v mém případě:
`<schema xmlns="http://www.ascc.net/xml/schematron">`
Pro správné hledání vzorů je potřeba doplnit namespace dokumentu OLIF.
`<ns prefix="o" uri="http://www.w3.org/2001/XMLSchema-instance" />`
Schematron využívá pro zápis vzorů jazyku *XPath*, elementů *assert* a *report*. Pro uschování dat se nastavuje v elementu *let* název proměnné a její hodnota.

- assert - vypíše určenou hlášku pokud není podmínka v atributu test splněna
- report - vypíše určenou hlášku pokud je podmínka v atributu test splněna
- let - proměnná pojmenovaná v *atributu* name a s hodnotou v atributu *value*

Tyto patří do elementu *rule*, který má v atributu *context* výraz určující, ke kterému prvku v dokumentu se vztahuje např. `"/"`, jež se vztahuje na všechny podelementy kořenového elementu. Všechny výše zmíněné značky patří do elementu *pattern*.

Ve svém předpisu schematronu využívám všechny výše zmíněné značky pro získání dat ze slovníku OLIF. Zjištění frází v dokumentu OLIFu, které nemají crLink je v příkladu 4.3.2, kdy hledám v elementech *entry*. Pokud tyto elementy obsahují *entryFormation* a zároveň neobsahují element *crLink* nebo hodnoty phr a headword, tak zahlásí report o frázi, která nemá headword.

Příklad 11 zjištění počtu frází v dokumentu OLIF

```
<pattern>
   <title>Fráze, které nemají headword</title>
   <rule context="o:entry">
      <let name="name" value="."/>
      <report test="
               descendant::o:entryFormation and
               (
                  not(descendant::o:crLink)
                  or not(descendant::o:entryFormation = 'phr')
                  or not(descendant::o:crLink = 'headword')
               )
                                 ">
Fráze, která nemá headword: <value-of select="$name" />
      </report>
   </rule>
</pattern>
```

Validace fixních hodnot OLIFu za pomoci schematronu

V dalším příkladu (viz 4.3.2) bude ukázáno, jak ověřuji fixní hodnoty, které mají obsahovat některé elementy. V tomto příkladu prohledávám hodnoty elementu *subjField* a pro dohledávání chyb si vždy každý záznam, ke kterému se daný *subjField* vztahuje ukládám do proměnné *name*. Dále v příkladu používám element *report* na zjištění toho zda hodnota neobsahuje ani jednu z fixních hodnot slovníku OLIF. Pokud tuto hodnotu neobsahuje, tak vypíšu hlášku záznam obsahující tuto nepovolenou hodnotu. Podobným způsobem je zpracované ověřování dalších fixních hodnot.

Příklad 12 validace správnosti fixních hodnot elementu subjField z 4.3.2.kapitoly

```
<pattern name="Element subjField obsahuje nepovolené hodnoty">
   <rule context="//o:subjField">
      <let name="name" value="parent::node()"/>
      <report
      test="(not(contains(.,'agriculture')) and not(contains(.,'audiovisual'))
      and not(contains(.,'aviation')) and not(contains(.,'botany/zoology'))
      and not(contains(.,'budget')) and not(contains(.,'chemistry'))
      and not(contains(.,'construction')) and not(contains(.,'customs'))
      and not(contains(.,'defense')) and not(contains(.,'development'))
      and not(contains(.,'economics')) and not(contains(.,'education'))
      and not(contains(.,'electrotechnics')) and not(contains(.,'employment'))
      and not(contains(.,'energy')) and not(contains(.,'environment'))
      and not(contains(.,'eurospeak')) and not(contains(.,'finance'))
      and not(contains(.,'fisheries')) and not(contains(.,'general'))
      and not(contains(.,'geology')) and not(contains(.,'industry'))
      and not(contains(.,'informatics')) and not(contains(.,'insurance'))
      and not(contains(.,'law')) and not(contains(.,'mechanics'))
      and not(contains(.,'medicine')) and not(contains(.,'mining'))
      and not(contains(.,'nuclear')) and not(contains(.,'social'))
      and not(contains(.,'statistics')) and not(contains(.,'steel'))
      and not(contains(.,'taxation')) and not(contains(.,'technology'))
      and not(contains(.,'telecom')) and not(contains(.,'trade'))
      and not(contains(.,'transport')))">
         subjField obsahuje nepovolenou hodnotu pro záznam:
         <value-of select="$name" />
      </report>
   </rule>
</pattern>
```

4.4 Problémy při implementaci a možnosti řešení

Získávání informací o existenci slov (jak anglických tak českých) ve slovnících jsem chtěl zahrnout do implementace parseru, ale vzhledem k časové náročnosti vykonávání českých slov voláním externího programu pro zjišťování existence českých slov v jiném slovníku jsem tuto část zahrnul do jiného Pythonovského skriptu. V implementaci parseru tak zbylo pouze ověření existence slov v anglickém slovníku (viz kapitola 4.2). Problém časové náročnosti volání externího programu napsaného jazykem C++ však přetrvává i v novém skriptu. V budoucnosti by bylo možnou cestou k vyřešení problému vytvořit pomocí programu SWIG modul pro python (verze SWIGu, se kterými jsem pracoval mi nevytvořil žádný modul schopný obstarat správnou funkčnost). Dalším problémem bylo zjišťení nejčastějšího slova z výsledného dokumentu na což mi

nestačila vyjadřovací síla jazyku XPath a Schematronu. Proto jsem opět použil skript v jazyce Python, kde vše proběhne rychle a bez problémů.

Kapitola 5

Statistiky

Statistiky získané ze slovníku ve formátu OLIF slouží k vyhodnocení základních informací o slovníku. Například počet záznamů ve slovníku, nejčastější slova v různých jazycích, slovo s největším počtem překladů, počet zkratek, počet frází a další. Ze statistik se dají odvodit chyby, které mohly nastat při implementaci parseru, např. když bych měl 200 000 slov v angličtině a veděl, že vstupní data mají informace o výslovnosti některých z nich a ve výsledných statistikách bych neměl ani jeden záznam s výslovností.

5.1 Implementace skriptu pro získání statistik

Pro získání statistik ze slovníku OLIF jsem použil skriptovací jazyk Python. Vzhledem k tomu, že výsledný slovník obsahuje velké množství dat a jazyk python podporuje parsovaní pomocí *sax*, tak jsem pro práci s dokumentem XML zvolil právě *sax* (viz kapitola 2.3.2), protože mi pro statistiky stačí data zpracovat jenom jednou a hledané hodnoty si uložit do proměnných a struktur pythonu. Pro hodnoty, kdy si ukládám slova a četnost jejich použití, používám pythonovskou strukturu Dictionary, což je asociativní pole záznamů. Po zpracování všech dat pole seřadím podle četnosti výskytu slov a vypíšu si tři první záznamy.

5.2 Výsledné statistiky

Výsledné statistiky pak zapisuji do dokumentu XML viz 5.2. Vysvětlení jednotlivých elementů a jejich atributů:

- entries - počet vstupů
- english - obsahuje anglická slova a jejich počet atribut count říká kolikrát je nalezeno
  - first - první slovo podle počtu výskytů v elementu canForm
  - second - druhé slovo podle počtu výskytů v elementu canForm
  - third - třetí slovo podle počtu výskytů v elementu canForm

- czech - obsahuje česká slova a jejich počet
  - first - první slovo podle počtu výskytů v elementu canForm atribut count říká kolikrát je nalezeno
  - second - druhé slovo podle počtu výskytů v elementu canForm atribut count říká kolikrát je nalezeno
  - third - třetí slovo podle počtu výskytů v elementu canForm atribut count říká kolikrát je nalezeno
- overal - obsahuje všechna slova a jejich počet
  - first - první slovo podle počtu výskytů v elementu canForm atribut count říká kolikrát je nalezeno
  - second - druhé slovo podle počtu výskytů v elementu canForm atribut count říká kolikrát je nalezeno
  - third - třetí slovo podle počtu výskytů v elementu canForm atribut count říká kolikrát je nalezeno
- trans - vztahuje se k překladům anglických slov do češtiny
  - words - obsahuje anglická slova a počet jejich překladů do češtiny atribut count říká kolikrát je nalezeno
    - first - první slovo podle počtu překladů
    - second - druhé slovo podle počtu překladů
    - third - třetí slovo podle počtu překladů
  - moretrans - obsahuje číslo, které říká kolik slov má více překladů
  - onetrans - kolik slov má právě jeden překlad
  - notrans - kolik slov nemá žádný překlad (např. zkratky, fráze bez překladu)
  - avgtrans - jaký je průměř překladů k anglickým slovům
- pronunciation - kolik anglických slov má výslovnost
- usages - kolik anglických a českých slov má příklad užití slova
- abbrevations - kolik je ve slovníku zkratek
- oth-variation - kolik je ve slovníku ortografických variant slova
- headwords - kolik slov má headword tzn. kolik je ve slovníku frází

Z výsledných statistik mě překvapil počet překladů slova *run*, které je přeloženo 851 krát.

```
<?xml version="1.0" encoding="utf-8"?>
<stats>
   <entries>220940</entries>
   <words>
   <english>
      <first count="714">go</first>
      <second count="624">take</second>
      <third count="474">get</third>
   </english>
   <czech>
      <first count="224">dát</first>
      <second count="147">vést</second>
      <third count="139">jít</third>
   </czech>
   <overal>
      <first count="715">go</first>
      <second count="624">take</second>
      <third count="474">get</third>
   </overal>
   </words>
   <trans>
      <words>
         <first count="851">run</first>
         <second count="848">set</second>
         <third count="693">go</third>
      </words>
      <moretrans>67123</moretrans>
      <onetrans>81863</onetrans>
      <notrans>60832</notrans>
      <avgtrans>2.09365891192</avgtrans>
   </trans>
   <pronunciation>83293</pronunciation>
   <usages>279687</usages>
   <abbrevations>14739</abbrevations>
   <orth-variation>1820</orth-variation>
   <headwords>126463</headwords>
</stats>
```

Ukázka 4: Výsledné statistiky ze slovníku OLIF

Kapitola 6

Závěr

Bakalářská práce se týká popisu, využití, nástrojů a výhod jazyku XML pro ukládání slovníkových dat do standardů k tomu určených. Popisuje různé XML standarty pro ukládání slovníkových dat a čerpání informací různých zdrojů, popisuje jejich odlišnosti a výhody.

Jedním z cílů práce byla transformace původních pseudo XML dat do jednoho ze standardů pro ukládání slovníkových dat. Za tento standard jsem si zvolil OLIF. Za účelem převodu jsem rozšířil implementaci již existujícího pythonovského parseru pro specifické datové vstupy. Zpracování všech vstupních dat bylo dle očekávání časově náročné.

Dále jsem se v práci zaměřil na pokročilou validaci XML dokumentu podle standardu OLIF a to pomocí schematronu. Konkrétně pak na validaci správnosti fixních hodnot, které má dokument OLIF obsahovat. Pro tento účel jsem si vytvořil předpis schematronu a aplikoval ho za pomoci již existujícího programu pro zpracování XSL transformací, saxonu. I tato část byla časově i paměťově náročná.

Poslední částí práce bylo zvolení vhodného postupu a získání statistik ze slovníku. K tomuto účelu jsem opět využil jazyku python a ke zpracování XML dokumentu parseru sax. I tato část je časově náročná avšak paměťově, díky saxu, ne.

Každá část práce požadovala specifické znalosti z oblasti zpracování XML dokumentů. Využívání či rozšiřování různých již implementovaných nástrojů i vytváření vlastních bylo pro mé znalosti znatelným přínosem.

Literatura

[1] Bray, T.; et al.: Extensible Markup Language (XML) 1.0 (Second Edition). [online], [cit. 1.4.2008].
URL <http://www.w3.org/TR/1998/REC-xml-19980210>

[2] Cimprich, P.: Akta X: Relax NG se prosazuje. [online], [cit. 19.4.2008].
URL <http://relaxng.org/spec-20011203.html>

[3] Consortium, O.: OLIF. [online], [cit. 19.4.2008].
URL <http://www.olif.net/>

[4] Corrigan, J.; Foster, T.: XLIFF: An Aid To Localization. [online], [cit. 2.5.2008].
URL <http://developers.sun.com/dev/gadc/technicalpublications/articles/xliff.html>

[5] Group, L. T. S. I.: TBX Specification. [online], [cit. 19.4.2008].
URL <http://www.lisa.org/fileadmin/standards/tbxISO_final.html>

[6] Ide, N.; Lenci, A.; Calzolari, N.: RDF Instantiation of ISLE/MILE Lexical Entries. 2003.
URL <citeseer.ist.psu.edu/article/ide03rdf.html>

[7] Initiaitve), T. T. E.: The Test Encoding Initiative Guidelines. [online], [cit. 19.4.2008].
URL
<http://etext.lib.virginia.edu/standards/tei/teip4/index.html>

[8] Kay, M. H.: SAXON. [online], [cit. 5.5.2008].
URL <http://www.blnz.com/xt/xt-20050823/index.html>

[9] Kosek, J.: XML pro každého. [online], [cit. 5.5.2008].
URL <http://www.kosek.cz/xml/>

[10] Lindsey, B.: XT. [online], [cit. 5.5.2008].
URL <http://www.blnz.com/xt/xt-20050823/index.html>

[11] LINGUISTICS, U. D. O. P. .: SAMPA computer readable phonetic alphabet. [online], [cit. 2.5.2008].
URL <http://www.phon.ucl.ac.uk/home/sampa/>

[12] Megginson, D.: SAX. [online], [cit. 11.4.2008].
URL <http://www.saxproject.org/about.html>

[13] RootsWeb.com: Abbreviations and Character Codes For RootsWeb.com Users. [online], [cit. 5.5.2008].
URL <http://helpdesk.rootsweb.com/codes/>

[14] Skonnard, A.; Gudgin, M.: *XML - pohotová referenční příručka*. Grada Publishing, a.s., 2006.

[15] for the Advancement of Structured Information Standards, T. O.: RELAX NG Specification. [online], [cit. 19.4.2008].
URL <http://relaxng.org/spec-20011203.html>

[16] Thurmair, G.: Exchange Formats: TBX, OLIF, and Beyond. [online], [cit. 19.4.2008].
URL <http://salome.coli.uni-bielefeld.de/gldv/2006_Heft1/45-56_Thurmair.pdf>

[17] W3C: Fixed Values for OLIF Data Categories. [online], [cit. 5.5.2008].
URL <http://www.olif.net/formalization/values/olifValuesFixJuly2001.xml>

[18] w3schools: DTD Tutorial. [online], [cit. 10.4.2008].
URL <http://www.w3schools.com/dtd/default.asp>

[19] w3schools: XML DOM(Document Object Model) Tutorial. [online], [cit. 10.4.2008].
URL <http://www.w3schools.com/dom/>

[20] w3schools: XML Information Set (Second Edition). [online], [cit. 10.4.2008].
URL <http://www.w3.org/TR/xml-infoset/>

[21] Wikipedia: Slovník. [online], [cit. 19.4.2008].
URL <http://cs.wikipedia.org/wiki/Slovn%C3%ADk>

[22] Zampolli, A.; Baroni, P.: International Standards for Language Engineering. [online], [cit. 2.5.2008].
URL <http://www.ilc.cnr.it/EAGLES96/isle/ISLE_Home_Page.htm>

Dodatek A

Přílohy bakalářské práce

Jako příloha bakalářské práce je jedno DVD s použitými nástroji.